In the Claims

1. **(Previously Presented)** A method for maintaining computer security comprising:

providing a signature file containing information about known system vulnerabilities, the information comprising a predefined length of a Universal Resource Locator ("URL") for a message header;

at a reverse proxy server residing between at least one client computer and a web server:

receiving an incoming message from the at least one client computer, wherein the incoming message, if malicious and upon receipt by the web server, automatically causes the web server to perform an action which exploits a vulnerability of the web server;

comparing a length of a URL in a message header of the incoming message ("the incoming URL") with the predefined length in the signature file to determine whether the incoming message is malicious; and

if the length of the incoming URL exceeds the predefined length, determining that the incoming message is malicious and blocking the incoming message from reaching the web server.

2. **(Previously Presented)** The method of claim 1, wherein the comparing further comprises:

parsing the incoming message;

converting the incoming message into an internal structure;

comparing the converted incoming message with the signature file; and

determining whether the converted incoming message is malicious based on the comparison.

3. **(Previously Presented)** The method of claim 2, further comprising reassembling the converted incoming message back into its original structure prior to forwarding it to the web server if it is determined that the code is not malicious.

4. **(Previously Presented)** The method of claim 1, further comprising forwarding the reassembled message to the web server if the length of the incoming URL is less than the predefined length.

5. **(Currently Amended)** The method of claim 1, wherein the signature file contains information about known vulnerabilities of ~~a client~~ the web server.

6. (**Original**) The method as claimed in claim 1, wherein the signature file is made available through a web server.

7. (**Original**) The method as claimed in claim 1, further comprising continuously updating the signature file.

8. (**Original**) The method as claimed in claim 1, further comprising periodically downloading the signature file in order to make its copy current.

9. **(Currently Amended)** A system for maintaining computer security comprising:

a signature file containing information about known system vulnerabilities, the information comprising a predefined length of a Universal Resource Locator ("URL") for a message header;

a web server; and

a tangible processor controlled device comprising a reverse proxy server residing between at least one client computer and ~~a~~ the web server, the reverse proxy server configured to:

receive an incoming message from the at least one client computer, wherein the incoming message, if malicious and upon receipt by the web server, automatically causes the web server to perform an action which exploits a vulnerability of the web server;

compare a length of a URL in a message header of the incoming message ("the incoming URL") with the predefined length in the signature file to determine whether the incoming message is malicious; and

if the length of the incoming URL exceeds the predefined length, determine that the incoming message is malicious and block the incoming message from reaching the web server.

10. **(Currently Amended)** The system of claim 9, wherein the reverse proxy server ~~machine~~ further comprises:

a Hypertext Transfer Protocol ("HTTP") message parser module for receiving, parsing and converting the incoming messages into a defined structure:

an HTTP message analyzer module for comparing the converted incoming messages with the signature file; and

an HTTP message reassembly module for reassembling the converted incoming messages determined not to be malicious into their original structure and forwarding them to the web server.

11. **(Previously Presented)** The system of claim 9, wherein the signature file contains information about known vulnerabilities of the web server.

12. **(Original)** The system of claim 9, wherein the signature file is made available through a web server.

13. **(Original)** The system of claim 9, wherein the signature file is continuously updated.

14. **(Currently Amended)** The system of claim 9, wherein the reverse proxy server ~~machine~~ periodically downloads the signature file in order to make its copy current.

15. **(Original)** The system of claim 10, wherein the signature file is linked to the HTTP message analyzer module.

16. **(Currently Amended)** A tangible computer storage medium including computer executable code for maintaining computer security, the computer executable code comprising:

code for accessing a signature file containing information about known system vulnerabilities, the information comprising a predefined length of a Universal Resource Locator ("URL") for a message header;

code for at a hypertext transfer protocol ("HTTP") reverse proxy server residing between at least one client computer and a web server:

receiving an incoming message from the at least one client computer, wherein the incoming message, if malicious and upon receipt by the web server, automatically causes the web server to perform an action which exploits a vulnerability of the web server;

comparing a length of a URL in a message header of the incoming message ("the incoming URL") with the predefined length in the signature file to determine whether the incoming message is malicious; and

if the length of the incoming URL exceeds the predefined length, determining that the incoming message is malicious and blocking the incoming message from reaching the web server.

17. **(Currently Amended)** The computer storage ~~recording~~ medium of claim 16, further comprising:

code for parsing the incoming message;

code for converting the incoming message into an internal structure;

code for comparing the converted incoming message with the signature file; and

code for determining whether the converted incoming message is malicious based on the comparison.

18. **(Currently Amended)** The computer storage ~~recording~~ medium of claim 17, further comprising code for reassembling the converted incoming message back into its original structure if it is determined that the code is not malicious.

19. (**Currently Amended**) The computer storage ~~recording~~ medium of claim 18, further comprising code for forwarding the reassembled message to the web server.

20. **(Currently Amended)** The computer storage ~~recording~~ medium of claim 16, wherein the signature file contains information about known vulnerabilities of ~~a client~~ the web server.

21. (**Currently Amended**) The computer storage ~~recording~~ medium of claim 16, wherein the signature file is made available through a web server.

22. (**Currently Amended**) The computer storage ~~recording~~ medium of claim 16, further comprising code for continuously updating the signature file.

23. **(Currently Amended)** The computer storage ~~recording~~ medium of claim 16, further comprising code for periodically downloading the signature file in order to make its copy current.

24. **(Previously Presented)** The method of claim 1, wherein the incoming message comprises a Hypertext Transfer Protocol ("HTTP") message.

25. **(Previously Presented)** The system of claim 9, wherein the incoming message comprises a Hypertext Transfer Protocol ("HTTP") message.

26. **(Previously Presented))** The computer storage medium of claim 16, wherein the incoming message comprises a Hypertext Transfer Protocol ("HTTP") message.

27. **(Canceled)**

28. **(Canceled)**

29. **(Canceled)**

30. **(Currently Amended)** The method of claim 1, wherein~~:~~ the information comprises a list of known system vulnerabilities; and further comprising:

comparing ~~the received incoming message with the signature file to determine whether the incoming message is malicious comprises determining whether the incoming message is malicious by determining whether~~ one or more characteristics of the incoming message with the list to determine whether the incoming message is malicious ; and

if the one or more characteristics fail to satisfy any ~~one~~ of the vulnerabilities on the list of known system vulnerabilities, determining that the incoming message is not malicious and forwarding the incoming message to the web server.

31. **(Currently Amended)** The system of claim 30~~-9~~, wherein at least one of the known system vulnerabilities is specific to the web server. ~~the viral signature patterns comprise one or more binary patterns associated with a virus.~~

32. **(Canceled)**

33. **(Previously Presented)** The method of Claim 1, wherein the incoming message is received from a first client computer; and further comprising:

if the incoming message is determined to be malicious, identifying the first computer; and

automatically blocking future messages received from the first client computer.

34. **(Previously Presented)** A method for maintaining computer security comprising:

providing a signature file containing information about known system vulnerabilities, the information comprising a predefined length of a Universal Resource Locator ("URL") for a message header;

receiving an incoming message from at least one client computer;

comparing a length of a URL in a message header of the incoming message ("the incoming URL") with the predefined length in the signature file to determine whether the incoming message is malicious; and

if the length of the incoming URL is greater than the predefined length, determining that the incoming message is malicious and blocking the incoming message from reaching a web server.

35. **(Previously Presented)** The method of Claim 34, wherein:

the predetermined length indicates a maximum amount of data that may be stored in a buffer of the web server before the buffer overflows;

the length of the incoming URL indicates an amount of data that the incoming message will attempt to store on the buffer if the incoming message is received by the web server; and

the step of determining that the incoming message is malicious comprises determining that the incoming message is capable of causing the buffer to overflow.